wksctl Documentation

wksctl development team

Feb 12, 2020

TABLE OF CONTENTS:

In this folder you can read more about how to use wksctl, and how it works:

CHAPTER

ONE

GET STARTED WITH `wksctl`

Using `wksctl` you have two modes of operation. **Standalone** mode and **GitOps** mode. The latter will enable you to keep the state of the cluster itself in Git too.

1.1 Modes of use

In **standalone mode**, `wksctl` builds a static cluster based on the contents of `cluster.yaml` and `machines.yaml` files passed on the command line; in **GitOps mode**, changes to `cluster.yaml` and `machines.yaml` files stored in Git will cause updates to the state of the live cluster.

1.1.1 Standalone mode

Run `wksctl apply` and pass in the paths to `cluster.yaml` and `machines.yaml`

```
wksctl apply \
  --cluster cluster.yaml \
  --machines machines.yaml
```

1.1.2 GitOps mode

We will create a cluster by pulling the cluster and machine yaml from git.

The following are new commandline arguments to `wksctl apply` which will result in a cluster being created.

- **git-url** The git repo url containing the `cluster.yaml` and `machine.yaml`
- **git-branch** The branch within the repo to pull the cluster info from
- **git-deploy-key** The deploy key configured for the GitHub repo
- **git-path** Relative path to files in Git (optional)

The git command line arguments will be passed instead of `--cluster` and `--machines`.

```
wksctl apply \
  --git-url git@github.com:$YOUR_GITHUB_ORG/config-repo.git \
  --git-branch dev \
  --git-deploy-key ./deploy-key
```

Using the url, branch, and deploy key, we will clone the repo - if we can't clone the repo we will error out.

These `--git` arguments are then used to set up and configure flux to automate cluster management.

We will rely on the user installing fluxctl to interact with flux directly instead of trying to replicate the functionality within `wksctl`

To see a more detailed example combining Wksctl, GitOps, Ignite also know as FireKube see Firekube.

CHAPTER

TWO

WKS AND FOOTLOOSE

1. Pick a distro of your choice:
    1. `centos7` (At the moment of writing, `centos7` is more mature)
    2. `ubuntu1804`

```
export DISTRO=centos7
```

2. Pick a backend of your choice:
    1. `docker` (not real VMs, but can be used on Mac)
    2. `ignite` (requires Ignite to be installed, and KVM functioning)

```
export BACKEND=docker
```

3. Install `footloose`:

```
GO111MODULE=on go install github.com/weaveworks/footloose
```

4. Start two machines using `footloose`:

```
$ footloose create -c ${DISTRO}/${BACKEND}/singlemaster.yaml
INFO[0000] Image: quay.io/footloose/centos7 present locally
INFO[0000] Creating machine: cluster-node0 ...
INFO[0001] Creating machine: cluster-node1 ...
```

You should now see the Container Machines running with `docker ps` or `ignite ps` (depending on your backend):

```
$ docker ps
CONTAINER ID        IMAGE                          COMMAND             CREATED
↪     STATUS              PORTS                                               NAMES
ab4f4b75f63d        quay.io/wksctl/vm-centos7 "/sbin/init"        5 seconds ago
↪       Up 4 seconds        0.0.0.0:2223->22/tcp, 0.0.0.0:6444->6443/tcp
↪cluster-node1
0ce280129e79        quay.io/wksctl/vm-centos7 "/sbin/init"        6 seconds ago
↪       Up 5 seconds        0.0.0.0:6443->6443/tcp, 0.0.0.0:2222->22/tcp
↪cluster-node0
```

or:

```
$ ignite ps
VM
↪ID                          IMAGE                                   KERNEL
```

(continues on next page)

(continued from previous page)

```
3fbe4611682b3e16        weaveworks/ignite-centos:latest        weaveworks/ignite-
↪kernel:4.19.47         4.0 GB         2         1024.0 MB         10m ago         Up␣
↪10m         172.17.0.3         0.0.0.0:30444->30443/tcp, 0.0.0.0:30081->30080/tcp,
↪ 0.0.0.0:2223->22/tcp, 0.0.0.0:6444->6443/tcp         centos-singlemaster-node1
b4fdde36eb122804        weaveworks/ignite-centos:latest        weaveworks/ignite-
↪kernel:4.19.47         4.0 GB         2         1024.0 MB         10m ago         Up␣
↪10m         172.17.0.2         0.0.0.0:2222->22/tcp, 0.0.0.0:6443->6443/tcp, 0.0.
↪0.0:30443->30443/tcp, 0.0.0.0:30080->30080/tcp         centos-singlemaster-node0
```

In case you would like to ssh into a machine e.g. `node0`, run:

```
footloose ssh -c ${DISTRO}/${BACKEND}/singlemaster.yaml root@node0
```

as the default user name is `root` for both backends.

5. Run `wksctl apply`:

```
wksctl apply \
 --machines=machines.yaml \
 --cluster=cluster.yaml \
 --verbose
```

6. Run `wksctl kubeconfig` to be able to connect to the cluster:

```
$ wksctl kubeconfig --cluster=cluster.yaml
To use kubectl with the example cluster, enter:
export KUBECONFIG=$HOME/.wks/weavek8sops/example/kubeconfig
$ export KUBECONFIG=/home/lucas/.wks/weavek8sops/example/kubeconfig
$ kubectl get nodes
NAME                STATUS   ROLES    AGE   VERSION
b4fdde36eb122804    Ready    master   77s   v1.14.1
$ kubectl get pods --all-namespaces
NAMESPACE     NAME                                    READY   STATUS    RESTARTS   AGE
kube-system   coredns-86c58d9df4-26gv9                1/1     Running   0          55s
kube-system   coredns-86c58d9df4-mb4h9                1/1     Running   0          55s
kube-system   etcd-13e2dc14bf30                       1/1     Running   0          6s
kube-system   kube-apiserver-13e2dc14bf30             1/1     Running   0          8s
kube-system   kube-proxy-l2fv7                        1/1     Running   0          55s
kube-system   kube-scheduler-13e2dc14bf30             1/1     Running   0          9s
kube-system   weave-net-n7lqb                         2/2     Running   0          55s
system        wks-controller-654d7cfb7c-47f9g         1/1     Running   0          54s
```

2.1 Multi-masters

Follow the above steps, but pass the multi-master manifests:

```
footloose create -c ${DISTRO}/${BACKEND}/multimaster.yaml
```

```
$ wksctl apply \
    --machines=machines-multimaster.yaml \
    --cluster=cluster.yaml \
    [...]
```

CHAPTER

THREE

WKS AND VAGRANT

```
$ make install
$ cd examples/vagrant
$ vagrant up
$ wksctl apply --cluster=cluster.yaml --machines=machines2.yaml --ssh-key=$HOME/.
↪vagrant/insecure_private_key
INFO[0000] installing CRI implementation
INFO[0054] installing Kubernetes
INFO[0079] initializing Kubernetes cluster with kubeadm
INFO[0164] installing CNI implementation
INFO[0165] applying cluster API's custom resource definitions
INFO[0166] applying cluster's manifest
INFO[0166] applying machines' manifest
INFO[0166] applying WKS controller's manifests
INFO[0167] adding SSH key to WKS secret and applying its manifest
$ vagrant ssh kube-01
$ sudo -i
# It takes a few minutes for the controller to create the second node.
# kubectl get nodes
NAME      STATUS   ROLES    AGE     VERSION
kube-01   Ready    master   8m8s    v1.13.3
kube-02   Ready    <none>   4m53s   v1.13.3

At any time, one can look at what the controller is doing:
# kubectl logs -n system wks-controller-6cf77fd8cc-wjpw2
```

CHAPTER

FOUR

WKS ON GCE

4.1 Prerequisites

1. Install:
   - `gcloud`
   - `jk`
   - `direnv`
2. Configure `gcloud`.
3. If you have never used direnv, you need to set it up. It hooks itself into your shell to set/unset environment variables when you enter or leave a directory.

```
eval "$(direnv hook bash)"
```

4. Create an `.envrc` file with, for example, the following content (or change to appropriate values where relevant):

```
export project="wks-tests"
export zone="europe-west1-b"
export user="$(whoami)"
export network="${user}-demo"
export image_project="centos-cloud"
export image_family="centos-7"
export network_interface="eth0"
```

   (see also: GCP images) and run:

```
direnv allow
```

4.2 Create the WKS cluster

1. Create the GCE network and instances:

```
./create-network.sh && ./create-instances.sh
```

   Note: In case cluster capacity for number of instances is reached, try using another zone in `.envrc`.

2. Generate `machines.yaml`:

```
./generate-machines-manifest.sh
```

3. Create & configure the WKS cluster:

```
wksctl apply
```

4. Generate the kubeconfig file

```
wksctl kubeconfig
```

This will print a line to the console similar to `$ export KUBECONFIG=/[path to kubeconfig file]`. Copy that line and paste it into any terminal you are using `kubectl` in.

5. Ensure that everything's working:

```
kubectl get namespaces
```

should return something like

```
NAME          STATUS   AGE
default       Active   1m59s
kube-public   Active   1m59s
kube-system   Active   1m59s
system        Active   1m54s
```

4.3 Delete the WKS cluster

To delete your WKS cluster, run:

```
./delete-instances.sh && ./delete-network.sh
```

CHAPTER

FIVE

FREQUENTLY ASKED QUESTIONS

5.1 Checkpoint and how to disable it

`wksctl` contacts Weaveworks servers for available versions. When a new version is available, `wksctl` will print out a message along with a URL to download it.

The information sent in this check is:

- wksctl version
- Machine Architecture
- Operating System
- Host UUID hash

To disable this check, run the following before executing `wksctl`:

```
export CHECKPOINT_DISABLE=1
```

CHAPTER

SIX

DEVELOPMENT

6.1 Build

```
make
```

6.1.1 Upgrading the build image

- Update `build/Dockerfile` as required.
- Test the build locally:

```
rm build/.uptodate
make !$
```

- Push this change, get it reviewed, and merge it to `master`.
- Run:

```
git checkout master ; git fetch origin master ; git merge --ff-only master
rm build/.uptodate
make !$
> [...]
> Successfully built deadbeefcafe
> Successfully tagged quay.io/wksctl/build:latest
> docker tag quay.io/wks/build quay.io/wksctl/build:master-XXXXXXX
> touch build/.uptodate
docker push quay.io/wksctl/build:$(tools/image-tag)
```

- Update `.circleci/config.yml` to use the newly pushed image.
- Push this change, get it reviewed, and merge it to `master`.

6.2 Adding docs

Docs live in the `docs` directory and we use Markdown for everything. Every new commit will be published at https://wksctl.readthedocs.io/en/latest/.

A few things to be aware of:

- The landing page (`docs/index.rst`) is written in reStructuredText, because readthedocs uses sphinx for generating HTML/PDF/etc. Make sure your new doc is listed there - it's how the index is built. (`recommonmark` is used as the bridge between Markdown and reStructuredText).
- Use `make serve-docs` to generate the docs locally and point a webbrowser to `localhost:8080` to check out if your changes worked out.
- Links in the docs will be automatically tested.
- Gotcha 1: links in markdown tables are problematic: https://github.com/ryanfox/sphinx-markdown-tables/issues/18
- Gotcha 2: cross-referencing using anchors is problematic: https://github.com/readthedocs/recommonmark/issues/8
- Gotcha 3: Make sure your use of headings, e.g. `#`, `##`, `###` makes sense. The table of contents will be a bit upset if you don't.